

## Modified Canny Edge Detection Algorithm

Notations:

$X_{raw}$  = raw RGB image

$X_{raw} \in \mathbb{R}^{3 \times M \times N}$

where  $M$  = number of rows,  $N$  = number of columns

$X_{raw}$  = raw pixel value of  $c^{th}$  color channel,  $i^{th}$  row,  $j^{th}$  column

$edge\_list \in \mathbb{R}^{num\_edges \times 2 \times 2}$

$edge\_list[e, v, i]$  = value of  $e^{th}$  edge,  $v^{th}$  vertex,  $i^{th}$  index

All arrays (matrices and tensors) are indexed from zero in this document.

## 1. Step 1: Gaussian Blur Filter:

### 1.1. Inputs:

1.1.1. Symmetric 2D Gaussian kernel matrix:  $G \in \mathbb{R}^{K \times K}$

1.1.1.1. Sample symmetric 2D Gaussian distribution:

$$G[x + p, y + p] = \frac{1}{2\pi\sigma^2} \exp(-(x^2 + y^2)/(2\sigma^2)), \quad \text{for } x, y \in \{-p, \dots, p\},$$

$$\text{where } p = (K - 1)/2$$

1.1.2. Normalize kernel (so that its elements sum to 1):

$$G[i, j] := G[i, j] / \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} G[i, j], \quad \text{for } i, j \in \{0, \dots, K - 1\}$$

1.1.3. Zero-padded raw image:  $\tilde{X}_{raw} \in \mathbb{R}^{3 \times (M+2p) \times (N+2p)}$

1.1.3.1. Zero-pad raw image for 2D “same” convolution:

$$\tilde{X}_{raw}[c, i, j] = \begin{cases} 0, & i < p \text{ or } j < p \text{ or } i > M - 1 + p \text{ or } j > N - 1 + p \\ X_{raw}[c, i - p, j - p], & \text{otherwise} \end{cases},$$

$$\text{for } c \in \{0, 1, 2\}, i \in \{0, \dots, (M + 2p) - 1\}, j \in \{0, \dots, (N + 2p) - 1\},$$

$$\text{where } p = (K - 1)/2$$

### 1.2. Computation:

1.2.1. Compute 2D “same” convolution:

$$X_{blur}[c, i, j] = \sum_{m=i}^{i+K-1} \sum_{n=j}^{j+K-1} \tilde{X}_{raw}[c, m, n] * G[m - i, n - j],$$

$$\text{for } c \in \{0, 1, 2\}, i \in \{0, \dots, M - 1\}, j \in \{0, \dots, N - 1\}$$

### 1.3. Outputs:

1.3.1. Blurred image:  $X_{blur} \in \mathbb{R}^{3 \times M \times N}$

## 2. Step 2: Gradient Estimation:

### 2.1. Inputs:

#### 2.1.1. Sobel operator kernels:

##### 2.1.1.1. Horizontal Sobel operator (for horizontal gradients):

$$S_{horiz} = \frac{1}{8} \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

##### 2.1.1.2. Vertical Sobel operator (for vertical gradients):

$$S_{vert} = \frac{1}{8} \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

#### 2.1.2. Blurred image: $X_{blur} \in \mathbb{R}^{3 \times M \times N}$

### 2.2. Computation:

#### 2.2.1. Convolve with horizontal and vertical Sobel operators (post zero-padded):

$$W_d[c, i, j] = \begin{cases} 0, & i = 0 \text{ or } j = 0 \text{ or } i = M - 1 \text{ or } j = N - 1 \\ \sum_{m=i-1}^{i+1} \sum_{n=j-1}^{j+1} X_{blur}[c, m, n] * S_d[m - (i - 1), n - (j - 1)], & \text{otherwise} \end{cases}$$

*for*  $d \in \{horiz, vert\}, c \in \{0, 1, 2\}, i \in \{0, \dots, M - 1\}, j \in \{0, \dots, N - 1\}$

#### 2.2.2. Compute L1-norm of horizontal and vertical gradients:

$$W[c, i, j] = |W_{horiz}[c, i, j]| + |W_{vert}[c, i, j]|,$$

*for*  $c \in \{0, 1, 2\}, i \in \{0, \dots, M - 1\}, j \in \{0, \dots, N - 1\}$

#### 2.2.3. Take maximum gradient over color channels:

$$Y[i, j] = \max_{c \in \{0, 1, 2\}} W[c, i, j], \quad \text{for } i \in \{0, \dots, M - 1\}, j \in \{0, \dots, N - 1\}$$

### 2.3. Outputs:

#### 2.3.1. Gradient image: $Y \in \mathbb{R}^{M \times N}$

### 3. Step 3: Horizontal Non-Maximal Suppression:

#### 3.1. Inputs:

3.1.1. Gradient Image:  $Y \in \mathbb{R}^{M \times N}$

3.1.2. (Half) suppression length:  $r$

#### 3.2. Algorithm:

```
for  $i = 0$  to  $M - 1$ :  
    for  $j = 0$  to  $N - 1$ :  
        if  $j \leq r - 1$ :  
            if  $Y[i, j] \neq \max_{j \leq k \leq j+r} Y[i, k]$ :  
                 $Y[i, j] := 0$   
        else if  $j \geq N - r$ :  
            if  $Y[i, j] \neq \max_{j-r \leq k \leq j} Y[i, k]$ :  
                 $Y[i, j] := 0$   
        else:  
            if  $Y[i, j] \neq \max_{j-r \leq k \leq j+r} Y[i, k]$ :  
                 $Y[i, j] := 0$ 
```

#### 3.3. Outputs:

3.3.1. Non-maximally suppressed gradient image:  $Y \in \mathbb{R}^{M \times N}$

#### 4. Step 4: Long Vertical Edge Determination:

##### 4.1. Inputs:

- 4.1.1. Non-maximally suppressed gradient image:  $Y \in \mathbb{R}^{M \times N}$
- 4.1.2. Low and high thresholds:  $thresh_{low}, thresh_{high}$
- 4.1.3. Vertical and horizontal “scanning” lengths:  $scan_{vert}, scan_{horiz}$
- 4.1.4. Minimum edge length:  $min\_edge\_length$

##### 4.2. Algorithm (Simplified):

```
num_edges = 0
for each pixel  $Y[i, j]$  in  $Y$ :
    if  $Y[i, j] \geq thresh_{high}$ :
        start =  $[i, j]$ 
        connect = true
        current_pix = start
        while connect == true and current_pix with scan lengths fits in  $Y$ :
             $max_{value} = \max_{i+1 \leq m \leq i+scan_{vert}, j-scan_{horiz} \leq n \leq j+scan_{horiz}} Y[m, n]$ 
             $max_{ind} = \arg \max_{i+1 \leq m \leq i+scan_{vert}, j-scan_{horiz} \leq n \leq j+scan_{horiz}} Y[m, n]$ 
            if  $max_{value} \geq thresh_{low}$ :
                current_pix =  $max_{ind}$ 
            else:
                connect = False
        stop = current_pix
        if  $|start[0] - stop[0]| + |start[1] - stop[1]| \geq min\_edge\_length$ :
            edge_list[num_edges] = [start, stop]
            num_edges += 1
```

##### 4.3. Outputs:

- 4.3.1. Array of edges:  $edge\_list \in \mathbb{R}^{num\_edges \times 2 \times 2}$